PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC1103- INTRODUCCIÓN A LA PROGRAMACIÓN

IIC1103 2015-2
Interrogación 2

Sábado 24 de octubre de 2015
Tiempo: 2 horas 30 minutos

Indicaciones

Lee completo el enunciado. La interrogación consta de tres preguntas. Sólo se reciben consultas sobre el enunciado durante los primeros 40 minutos. Los ayudantes no están autorizados a responder preguntas. La última página contiene un recordatorio; no tienes permitido usar material de apoyo adicional. A menos que el enunciado indique lo contrario, en tus respuestas no necesitas validar que el usuario haya ingresado los datos de manera correcta.

Problema 1 (60 puntos)

Un taller mecánico trabaja en vehículos, los cuales tienen una marca, un modelo, una patente, y un propietario (que tiene nombre y teléfono). Los vehículos pueden estar en uno de los siguientes estados: en solicitud de presupuesto (P), en diagnóstico por un mecánico para elaborar el presupuesto (D), siendo arreglado (A), en espera de entrega (E), o retirado cuando el propietario ya se lo llevó (R). Debes implementar lo siguiente:

a) (5 puntos) La clase `Vehiculo`, con los atributos marca, modelo, patente y propietario.

b) (5 puntos) La clase `Taller`, que posee un único atributo correspondiente a una lista de vehículos, y en la que debes implementar los siguientes 3 métodos:

   I) (10 puntos) `recibir_vehiculo(self, vehiculo)` : agrega `vehiculo` al taller, en estado de solicitud de presupuesto.

   II) (20 puntos) `actualizar_estado(self, patente, nuevo_estado)`: cambia el estado del vehículo de la `patente` dada al `nuevo_estado` señalado. Puedes asumir que el vehículo con la patente indicada existe en el taller.

   III) (20 puntos) `entrega_vehiculos(self)`: entrega una lista con los nombres y teléfonos de los propietarios de vehículos que están a la espera de ser entregados. Cada elemento de la lista que entrega la función debe ser una lista compuesta de dos strings (nombre y teléfono).

   Contempla en tu solución guardar un estado por cada vehículo.

Problema 2 (60 puntos)

Un lipograma inverso de una letra minúscula ℓ es un texto en el que cada palabra contiene a ℓ o a la mayúscula de ℓ. Por ejemplo, el texto "**T**e fal**t**a **t**an**t**a ac**t**i**t**ud" es un lipograma inverso de **t** porque cada palabra contiene a **t** o a **T**. Puedes suponer que todas las letras son sin tildes, las palabras están separadas sólo por espacios y que no hay signos de puntuación.

a) (40 puntos) Implementa una función `lipograma_inverso(frase)`, que retorne una letra de la cual es lipograma inverso el string `frase`, o `"0"` si no es lipograma inverso de ninguna letra.

b) (20 puntos) Escribe un programa que analice frases contenidas en el archivo `frases.txt`, donde cada línea del archivo contiene una frase que debe ser analizada para determinar si es o no un lipograma inverso, y en caso de que sí sea, de qué letra. La decisión debe almacenarse en un nuevo archivo `salida.txt`, donde por cada línea de `frases.txt` debe almacenarse una línea con la decisión. Si se determina que alguna frase es lipograma inverso de más de una letra, puedes usar cualquiera de ellas.

Ejemplo:

Archivo `frases.txt`

```
Te falta tanta actitud
Tengo mucho sueño
Voy a correr
frutillas fresas frambuesas
```

Archivo `salida.txt`

```
Lipograma inverso de t
Lipograma inverso de o
No es lipograma inverso
Lipograma inverso de f
```

Problema 3 (60 puntos)

En Chiaolún, un tradicional pueblo chino, aún se acostumbra a que mujeres sólo bailan con hombres, y viceversa. Este pueblo tiene muchas más mujeres que hombres y por lo tanto se vuelve complejo organizar una fiesta. Esto es porque allí a las mujeres les gusta mucho bailar y se considera culturalmente inaceptable que alguien que quiere bailar se quede sin hacerlo. Se complica aún más la situación porque a los hombres no les gusta bailar demasiado. De hecho, cada hombre tiene un límite de bailes que está dispuesto a bailar en una fiesta.

El objetivo de esta pregunta es que escribas una función para dar un "plan de baile", si éste existe, para una fiesta en Chiaolún dada una descripción de los invitados. Deberás escribir una función `plan` que reciba dos listas: una con los nombres de las mujeres asistentes a la fiesta y otra con pares de la forma `[n,c]` donde `n` es el nombre de un hombre y `c` un número entero positivo, que representa la cantidad máxima de bailes que está dispuesto a bailar. Tu función deberá retornar una lista de pares `[h,m]`, donde `h` es el nombre de un hombre y `m` es el nombre de una mujer, expresando el hecho de que "`h` baila con `m`". La lista debe ser tal que:

- todas las mujeres bailan, y
- ningún hombre baila más de lo que está dispuesto a bailar.

No importa si en el plan retornado algún hombre no baila (a ellos realmente no les llama la atención el baile). La función deberá retornar una lista vacía si es que no existe un plan de baile (es decir, cuando no es posible que **todas** las mujeres bailen). Por ejemplo, al llamar tu función de esta manera:

```
plan(['siugmin','daiyu','fenfang'], [['xiaoxun',1], ['shitong',2]])
```

ésta podría retornar:

```
[['xiaoxun','siugmin'],['shitong','daiyu'],['shitong','fenfang']].
```

Sin embargo el llamado:

```
plan(['siugmin','daiyu','fenfang','jiaying'], [['xiaoxun',1], ['shitong',2]])
```

deberá retornar `[]`.

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC1103- INTRODUCCIÓN A LA PROGRAMACIÓN

Recordatorio contenidos I2 - Segundo semestre 2015

En los ejemplos, lo marcado como `<texto>` se interpreta como lo que debes rellenar con tu código según cada caso.

1. Tipos de datos y operadores

Tipo de dato	Clase	Ejemplo
Números enteros	`int`	`2`
Números reales	`float`	`2.5`
Números complejos	`complex`	`2 + 3j`
Valores booleanos	`bool`	`True`/`False`
Cadenas de texto	`str`	`"hola"`

Operación	Descripción	Ejemplo
`+`	Suma	`2.3+5.4`
`-`	Resta	`45.45-10.02`
`-`	Negación	`-5.4`
`*`	Multiplicación	`(2.3+4.2j)*3`
`**`	Potenciación	`2**8`
`/`	División	`100/99`
`//`	División entera	`100//99`
`%`	Módulo	`10%3`

Prioridad (de mayor a menor): `()`; `**`; `*`, `/`, `//` o `%`; `+` o `-`.

Operación	Descripción	Ejemplo
`==` `(!=)`	Igual (distinto) a	`2==2`
`<` `(<=)`	Menor (o igual)	`1<1.1`
`>` `(>=)`	Mayor (o igual)	`3>=1`
`and`	Ambos `True`	`2>1 and 2<3`
`or`	Algún `True`	`2!=2 or 2==2`
`not`	Negación	`not True`

Prioridad (de mayor a menor): `()`; `or`; `and`; `not`; comparadores.

2. Funciones predefinidas

- `int(arg)` convierte `arg` a entero.
- `float(arg)` convierte `arg` a número real.
- `str(arg)` convierte `arg` a cadena de texto (string).
- `list(arg)` genera una lista con elementos según `arg`, que debe ser *iterable* (strings, listas, tuplas, `range`).

3. Función `print`

- Un argumento:
  `print(arg)`
- Dos o más argumentos:
  `print(arg1, arg2,arg3)`
- Uso de parámetros, por ejemplo, para eliminar salto de línea y separar con guión:
  `print(arg, sep='-', end='')`

4. Función `input`

- `ret = input(texto)` guarda en `ret` un `str` ingresado.
- `ret = int(input(texto))` guarda en `ret` un `int` ingresado.
- `ret = float(input(texto))` guarda en `ret` un `float` ingresado.

5. `if/elif/else`

```
if <cond 1> :
    <codigo si se cumple cond 1>
    if <cond 1.1> :
        <codigo si se cumple 1.1>
    else :
        <codigo si no se cumple 1.1>
elif <cond 2> :
    <codigo si se cumple cond 2 pero no cond 1>
else :
    <codigo si no se cumple cond 1 ni cond 2>
```

6. `while`

```
while <condicion> :
    <codigo que se ejecuta repetidas
      veces mientras se cumpla
      condicion>
```

7. Funciones propias

```
def funcion(<argumentos>):
    <codigo de funcion>
    return <valor de retorno>
```

Variables y parámetros definidos dentro de funciones no son visibles fuera de la función (*scope* local).

8. Programación orientada a objetos

```
class <NombreClase>:
    def __init__(self, <parametros>):
        self.<atributos> = <algun parametro>
    def __str__(self):
        <codigo sobrecarga de funcion str()>
        return <string que representa
                        los atributos>
    def <metodo propio>(self, <parametros>):
        <codigo de modulo propio>
```

9. Strings, clase `str`

Acceso a caracteres particulares con operador `[]`, partiendo con índice cero. Porción de string con *slice*, por ejemplo si `string='Hola'`, `string[1:3]` es `'ol'`. Algunos métodos y funciones de strings:

- Operador `+`: une (concatena) dos strings.
- Operador `in`: cuando `a in b` retorna `True`, entonces el string `a` está contenido en el string `b`.
- `string.find(a)`: determina si `a` está contenido en `string`. Retorna la posición (índice) dentro de `string` donde comienza la primera aparición del sub-string `a`. Si no está, retorna `-1`.
- `string.upper()`, `string.lower()`: retorna `string` convertido a mayúsculas y minúsculas, respectivamente.
- `string.strip()`: retorna un nuevo string en que se eliminan los espacios en blanco iniciales y finales de `string`.
- `string.split(a)`: retorna una lista con los elementos del `string` que están separados por el string `a`. Si se omite `a`, asume que el separador es uno o más espacios en blanco o el salto de línea.
- `p.join(lista)`: suponiendo que `p` es un string, retorna un nuevo string conteniendo los elementos de la lista "unidos" por el string `p`.
- Función `len(string)`: entrega el número de caracteres de `string`.

Una forma de iterar sobre los caracteres de `string`:

```
for char in string:
    <operaciones con el caracter char>
```

10. Listas

Secuencias de elementos-objetos. Se definen como `lista = [<elem1>,<elem2>,...,<elemN>]`. Los elementos pueden o no ser del mismo tipo. Para acceder al elemento `i`, se usa `lista[i]`. La sublista `lista[i:j]` incluye los elementos desde la posición `i` hasta `j-1`. Algunos métodos y funciones de listas:

- Operador `+`. concatena dos listas.
- Operador `in`: `a in b` retorna `True` cuando el elemento `a` está contenido en la lista `b`. Si no está contenido, retorna `False`.
- `lista.append(a)`: agrega `a` al final de la lista.
- `lista.insert(i,a)`: inserta el elemento `a` en la posición `i`, desplazando los elementos después de `i`.
- `lista.pop(i)`: retorna el elemento de la lista en la posición `i`, y lo elimina de `lista`.
- `lista.remove(elem)`: elimina la primera aparición de `elem` en la lista.
- Función `len(lista)`: entrega el número de elementos de `lista`.

Para iterar sobre los elementos de `lista`:

```
for elem in lista:
    <lo que quieran hacer con elem>
```

11. Archivos

Abrir un archivo:
`archivo = open(<nombre archivo>,<modo>)`,
p. ej. `archivo = open('archivo.txt','r')`.
`<modo>` puede ser `'w'` para escribir un archivo nuevo, `'r'` para leer un archivo (predeterminado), y `'a'` para escribir en un archivo ya existente, agregando datos al final del archivo.

Algunos métodos del objeto que retorna la función `open`:

- `archivo.readline()`: retorna un string con la línea siguiente del archivo, comenzando al inicio del archivo.
- `archivo.write(string)`: escribe en el archivo el string `string`.
- `archivo.close()`: cierra el archivo.

Para leer un archivo entero puedes usar `for`, que iterará línea por línea del archivo:

```
archivo = open('archivo.txt','r')
for linea in archivo:
    <lo que quieran hacer con linea>
```